**Sustainable Smart City Assistant AI**

**Project Documentation**

# 1.Introduction

* Project tile: Suistanable Smart City Assistant Ai
* Team members :B.Divasagar,G.Ravivarman,D.Sudesh,Vignesh **Overview :**

The **Sustainable Smart City Assistant AI** is a web-based interactive tool designed to support environmentally-conscious citizens, policy makers, and urban planners by providing:

* ✅ **Eco-Friendly Living Tips** based on environmental challenges or keywords.
* 📑 **Policy Document Summarization**, extracting actionable insights and summarizing key points from uploaded PDFs or text-based inputs.

Powered by **IBM Granite 3.2-2B Instruct**, the assistant uses large language models (LLMs) to generate relevant content that supports **green living**, **sustainability**, and **smart urban governance**.

**Future :**

**🔮 Future of the Project**

The **Sustainable Smart City Assistant AI** has a strong foundation as a tool for sustainability, eco-awareness, and policy comprehension. Moving forward, the project can evolve into a full-featured, intelligent assistant for **smart city development, climate action, and civic engagement**. Below are key directions for its future:

**🌐 1. Expand Use Cases Beyond Eco Tips & Policies**

* **Urban Planning Insights**: Integrate AI to suggest city infrastructure improvements (e.g., public transport, green spaces, traffic flow).
* **Civic Services Assistant**: Help citizens navigate local sustainability programs, waste disposal rules, or energy incentives.
* **Sustainability Score Generator**: Automatically score and suggest improvements for cities, policies, or projects based on SDG (Sustainable Development Goals) criteria.

**📡 2. Real-Time Smart City Data Integration**

* Integrate IoT data streams (e.g., pollution sensors, traffic cameras, energy usage).
* Provide **live sustainability dashboards** using AI to interpret data.
* Enable **predictive alerts** (e.g., water shortages, energy spikes, heat waves).

**🧠 3. AI Model Enhancements**

* **Fine-Tuning** the model on domain-specific data:
  + Climate action reports
  + City planning documents
  + Sustainability assessments
* Support **multilingual capabilities** for global city applicability.
* Add **vision-language models** to analyze policy documents with graphs, tables, or diagrams.

**📱 4. Mobile & Web App Development**

* Convert the Gradio prototype into a **cross-platform app** (Flutter, React Native, or Progressive Web App).
* Add user personalization (e.g., eco-habit tracker, local policy notifications).
* Use voice interaction via speech-to-text (for accessibility).

**📚 5. Educational Integration**

* Build an **AI-powered sustainability curriculum assistant** for schools and universities.
* Create interactive case studies and simulations for urban development and policy design.

**🏛️ 6. Partnerships & Open Data Platforms**

* Collaborate with:
  + Local governments
  + Environmental NGOs
  + UN Smart Cities initiatives
* Integrate open city data portals for more contextual outputs.

**🛡️ 7. Ethics, Transparency & Trust**

* Add **explainable AI (XAI)** features:
  + Show sources or reasoning behind suggestions/summaries.
* Implement **bias audits** to ensure fairness in policy summarization.
* Provide options to **flag or validate** AI outputs with expert human review.

**💡 8. Crowdsourced Green Innovation**

* Allow users to submit eco-tips and verify effectiveness.
* Create a **community-driven tips library** moderated by AI and sustainability experts.

# 4. Setup Instructions

Prerequisites:

* Python 3.9+
* Google Colab account
* Hugging Face access

Installation (in Colab):

**!pip install transformers torch gradio pandas plotly -q** Run the app:

**Demo.queue()**

**Demo.launch(share=True)**

# 5. Folder / Notebook Structure

Healthai.ipynb

│

├── Model Loader (with fallback)

├── Guardrails

├── Disease Prediction

├── Treatment Plan

├── Chatbot

├── Analytics Dashboard

└── Gradio UI

# 6. Running the Application

1. Open Google Colab.
2. Upload and run healthai.ipynb.
3. Install dependencies.
4. Launch the Gradio interface.
5. Use tabs to test disease prediction, treatment plans, chatbot, and analytics.

# 7.API Documentation

Backend APIs available include:

POST /prediction/analyze

* Accepts patient profile and symptoms
* Returns possible conditions, red flags, and general recommendations in JSON

POST /treatment/plan

* Accepts patient profile and a medical condition
* Responds with a general treatment overview, lifestyle tips, and when to seek care

POST /chat/ask

* Accepts a user health-related query
* Responds with an AI-generated safe and friendly answer in bullet points

POST /analytics/upload

* Accepts CSV or Excel files containing health vitals (Heart Rate, BP, Glucose, etc.)
* Returns an interactive Plotly chart showing weekly health trends

GET /disclaimer

* Returns the medical disclaimer used in all responses

# 8. Authentication

**🔐 Authentication for Smart City Assistant AI**

**🔸 Why Add Authentication?**

Authentication allows you to:

* Secure access to certain features (e.g., admin dashboard, user preferences)
* Save user-specific data like eco-tips history or summaries
* Protect against misuse or API abuse
* Support role-based access (e.g., citizen vs. city planner)

# 9.User Interface (Gradio)

# The app interface is built using Gradio's Blocks and Tabs, allowing an intuitive UX:

# Tabs:

# 🌱 Eco Tips Generator: Input keywords → Get green tips

# 📜 Policy Summarizer: Upload or paste policy → Get summary

# Buttons:

# Generate Eco Tips ✅

# Summarize Policy 📑

# 10. Testing

Unit Testing → Prompt builders, JSON parsing

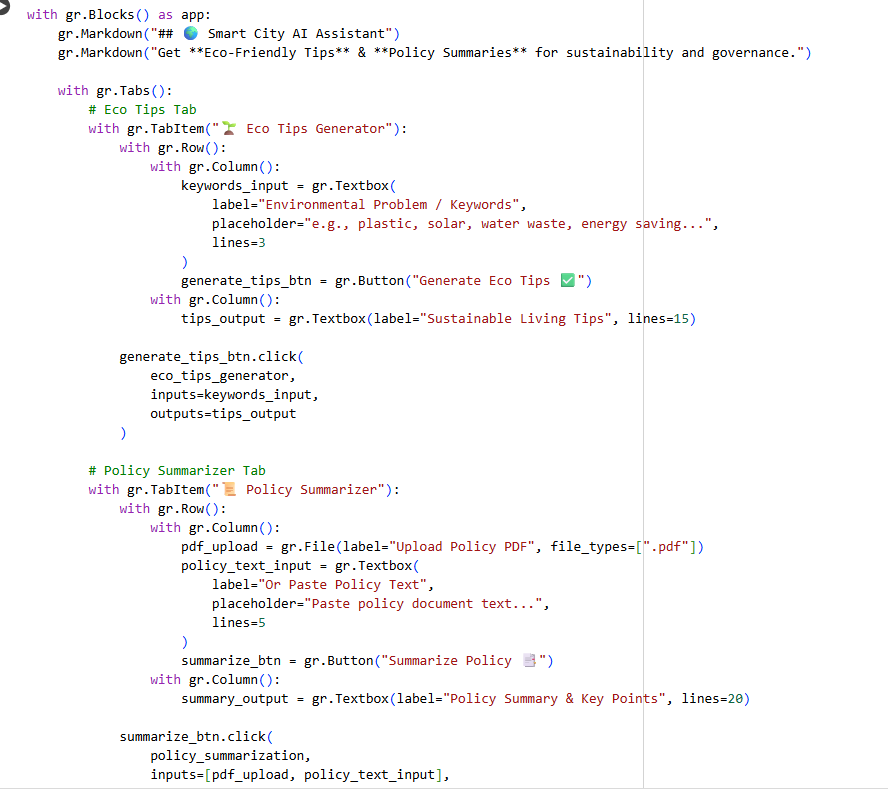
Manual Testing → Chatbot answers, prediction/treatment accuracy

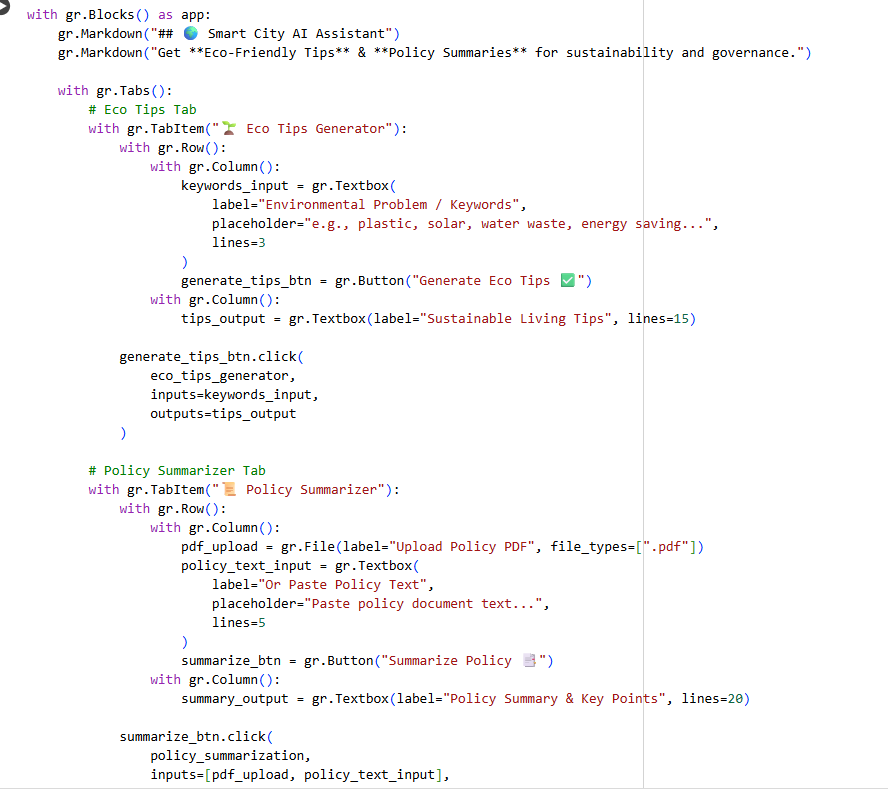
Edge Cases → Empty inputs, invalid CSV files

# 11. Screenshots

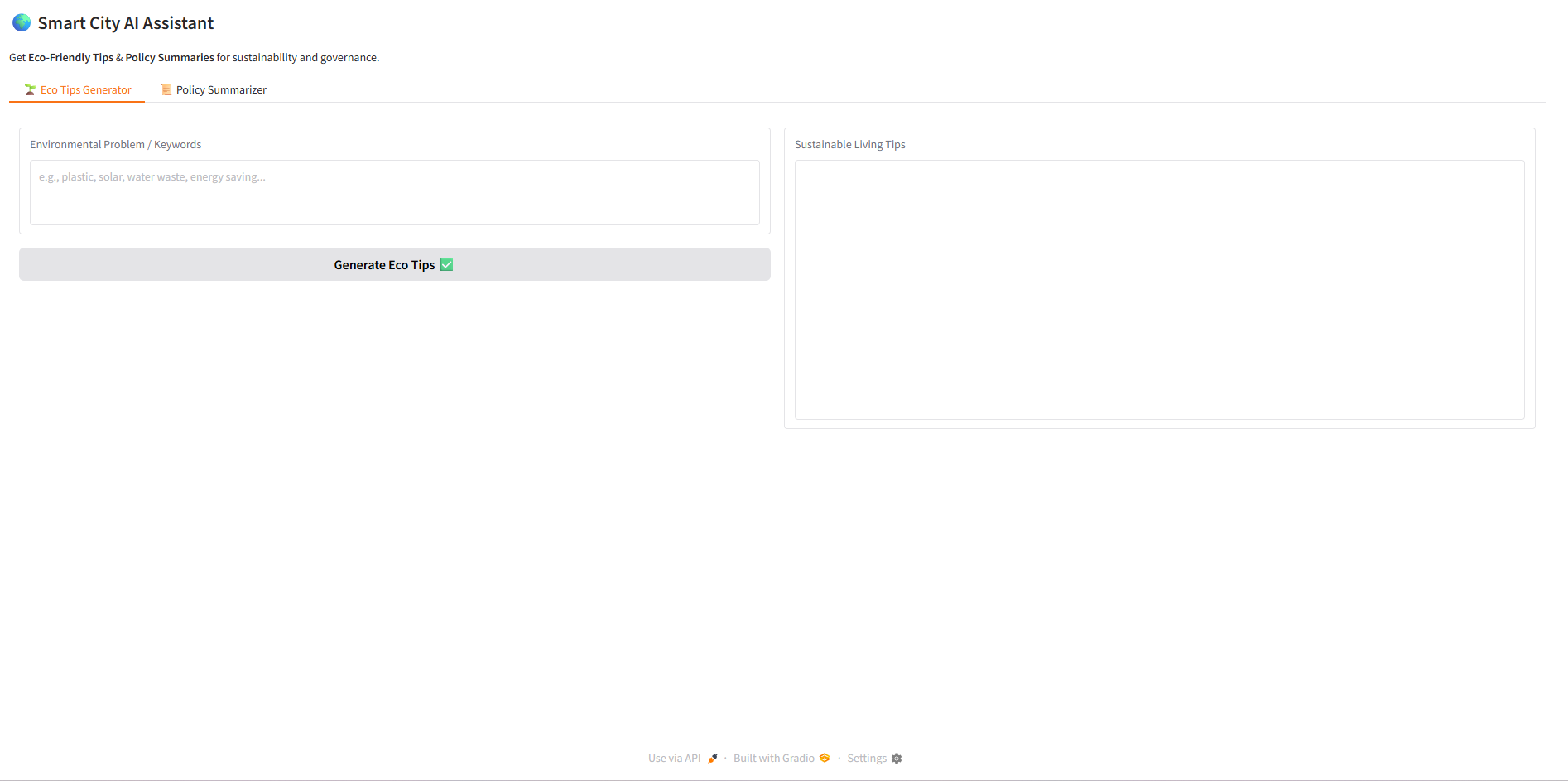
Program :

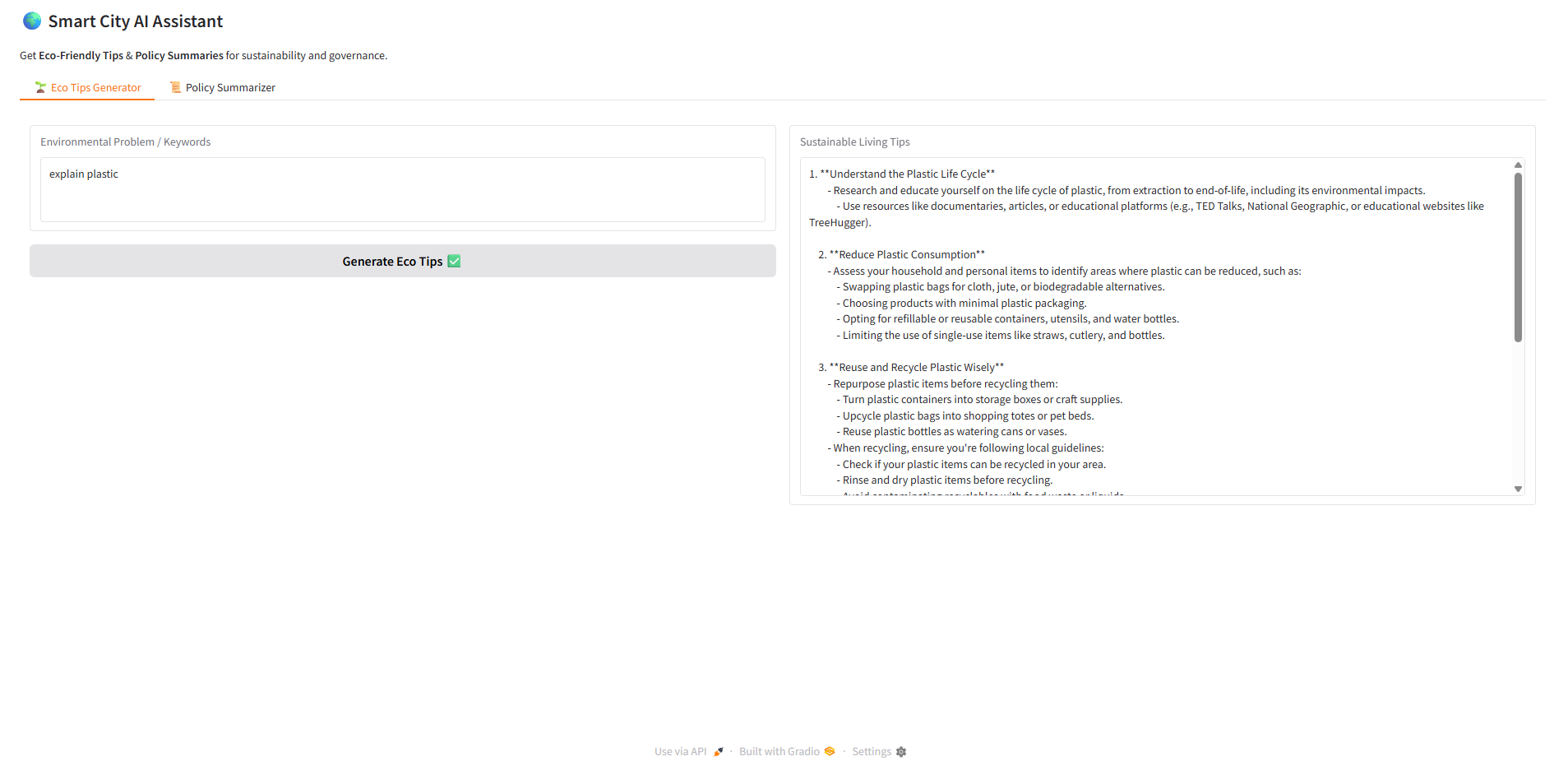






Output :





12. Future Enhancements

**🔮 Future Enhancements Roadmap**

**💡 1. User Authentication & Profiles**

* **Login/Signup** system (e.g., via Google, Firebase)
* Personalized dashboard for:
  + Saving past queries and summaries
  + Submitting feedback
  + Viewing recommended actions based on user behavior